Tampereen yliopisto

Amortisoitu tehokkuus ja std::vector:n muistinhallinta

COMP.CS.300 Tietorakenteet ja algoritmit 1

Matti Rintala (matti.rintala@tuni.fi)

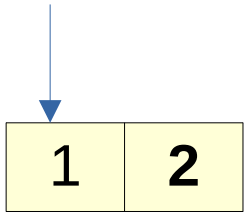
Tampereen yliopisto

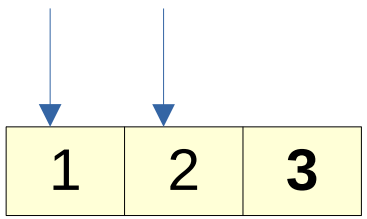
STL:n vektorin muistinhallinta

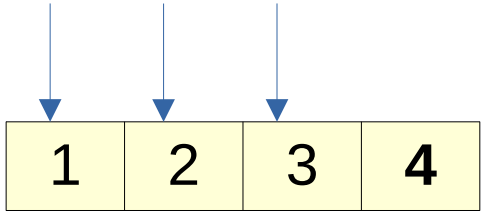
- Vektori varaa yhtenäisen muistialueen alkioilleen
- Mitä tehdä, kun tarvitaan lisää tilaa?

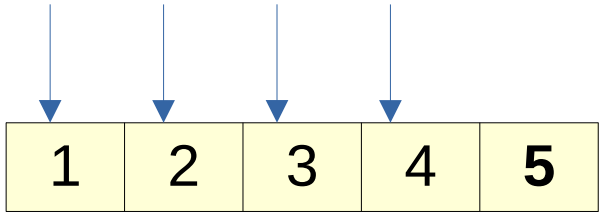
1

- Yritys 1: varataan tarvittavan verran suurempi muistialue, kopioidaan vanhat alkiot sinne


1
2


1
2
3

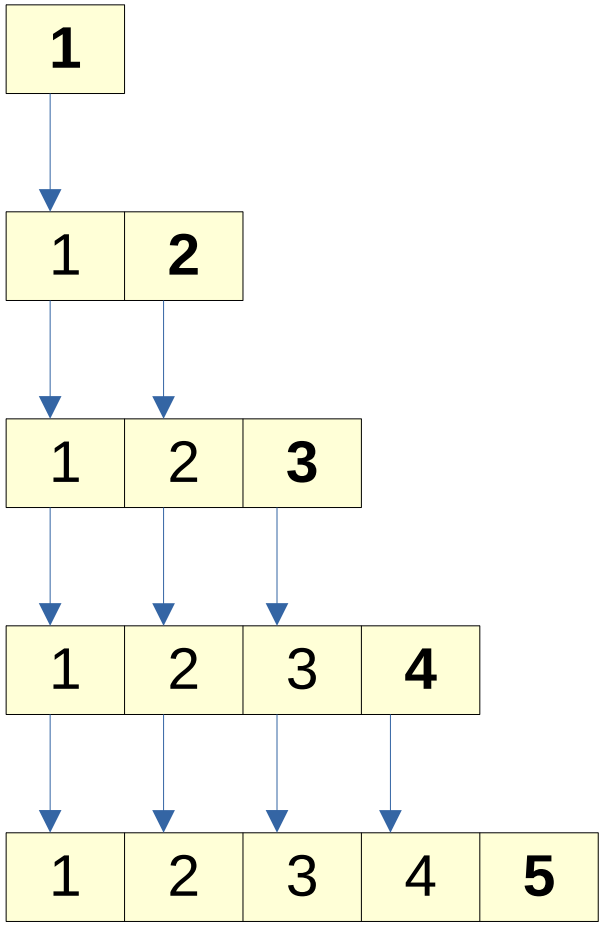

1
2
3
4


1
2
3
4
5

Tampereen yliopisto

STL:n vektorin muistinhallinta

- Vektori varaa yhtenäisen muistialueen alkioilleen
- Mitä tehdä, kun tarvitaan lisää tilaa?
- Yritys 1: varataan tarvittavan verran suurempi muistialue, kopioidaan vanhat alkiot sinne



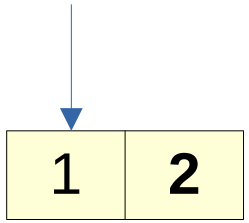
Tampereen yliopisto

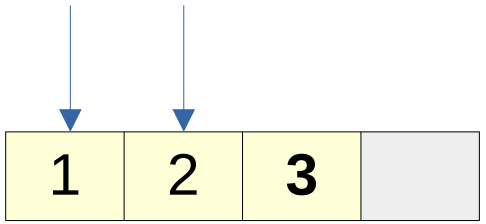
STL:n vektorin muistinhallinta

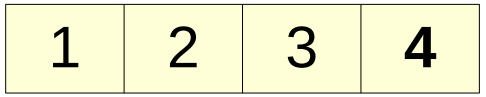
- Vektori varaa yhtenäisen muistialueen alkioilleen
- Mitä tehdä, kun tarvitaan lisää tilaa?

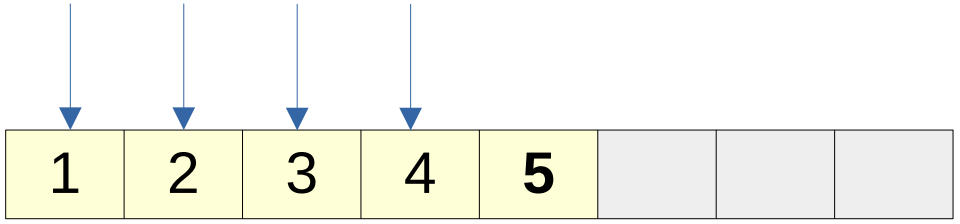
1

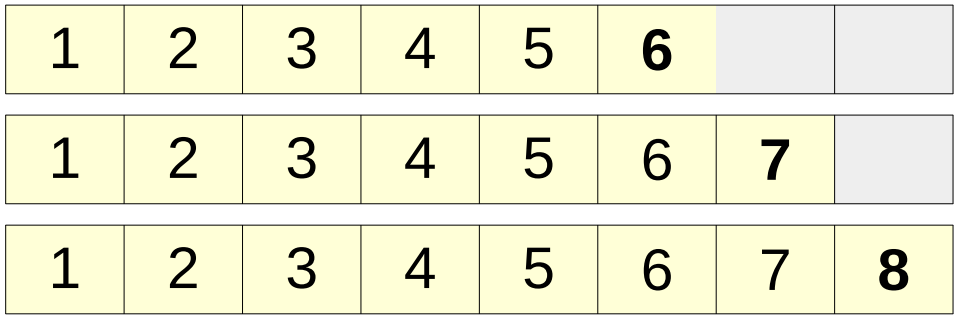
- Yritys 2: varataan **kaksi kertaa** vanhan kokoinen muistialue, kopioidaan vanhat alkiot sinne
- Huom: Osa lopusta jää vielä käyttämättä!


1
2


1
2
3


1
2
3
4

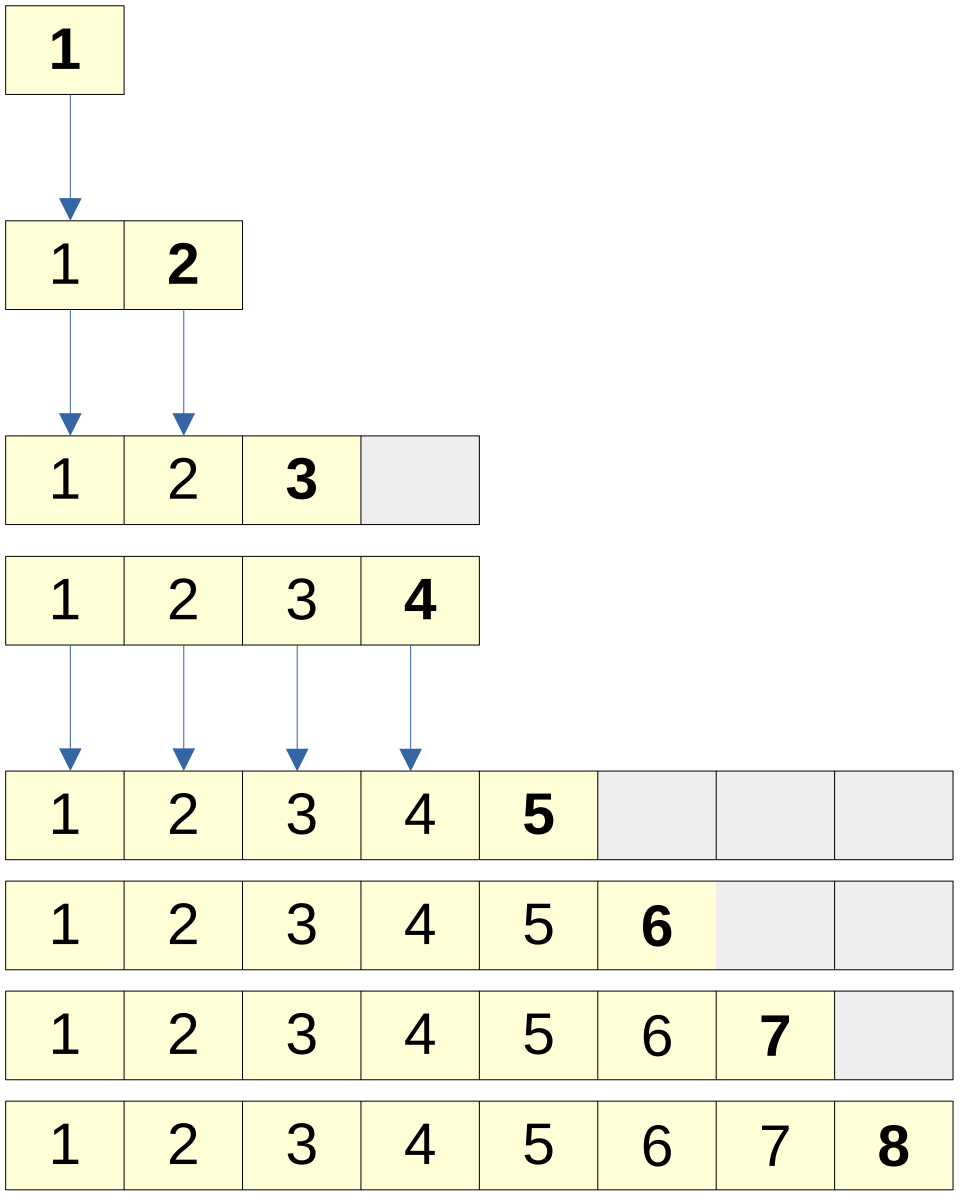

1
2
3
4
5


1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8

Tampereen yliopisto

STL:n vektorin muistinhallinta

- Vektori varaa yhtenäisen muistialueen alkioilleen
- Mitä tehdä, kun tarvitaan lisää tilaa?
- Yritys 2: varataan **kaksi kertaa** vanhan kokoinen muistialue, kopioidaan vanhat alkiot sinne
- Huom: Osa lopusta jää vielä käyttämättä!



Amortisoitu tehokkuus

- "Tasattu (asymptoottinen) tehokkuus"
- Lasketaan *operaatiosarjojen* keskimääräinen tehokkuus
- Kalliin harvinaisen operaation kustannus voidaan jakaa tasan halvoille
- Esim. vektoriin lisäys:
  - Yksittäinen vektorin lisäys voi olla lineaarinen (yhä harvinaisempaa)
  - Lisäykset silti *amortisoidusti* vakioaikaisia (keskimäärin)

std::vector

- STL:n vektori sisältää operaatioita muistinhallinnan säätämiseen
- vec.reserve(n): Varaa *muistia* väh. n alkiolle, alkioita ei silti (vielä) lisätä
- vec.capacity(): Alkioiden maksimimäärä ilman uutta muistinvarausta
- vec.shrink_to_fit(): Siirrä alkiot muistialueeseen, joka on juuri sopivan kokoinen
- (vec.erase() *ei* vapauta muistia!)